Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра компьютерных систем в управлении и проектировании (КСУП)

**СТРУКТУРЫ И ПЕРЕЧИСЛЕНИЯ**

Отчёт по лабораторной работе №2 по дисциплине

«Объектно-ориентированное программирование»

Студент группы 588-1

\_\_\_\_\_\_\_\_\_\_ / Чан Хыу Тхай

«\_» \_\_\_\_\_\_\_\_\_2022 г.

Руководитель старший научный

сотрудник, доцент каф. КСУП

\_\_\_\_\_\_\_\_\_ / Горяинов А.Е.

«\_\_\_» 2022 г.

Томск 2022

1. **Цель работы**

Цель этой лабораторной работы — ознакомиться со структурами и типами enum и получить практические навыки решения связанных с ними задач в объектно-ориентированном программировании.

1. **Теоретические основы**

**Структура struct** – очень интересный тип данных: это набор переменных разного типа, объединённых одним именем. В некоторых случаях структуры позволяют сильно упростить написание кода и сделать его более понятным, а также упростить придумывание новых имён для переменных. А ещё структура – это практически класс (урок про классы)! Но без механизмов наследования и приватных-публичных членов.

Структура объявляется по следующей схеме:

struct ярлык {

тип\_данных имя\_переменной\_1;

тип\_данных имя\_переменной\_2;

тип\_данных имя\_переменной\_3;

};

Ярлык будет являться новым типом данных. Используя этот ярлык, можно объявлять уже непосредственно саму структуру как переменную:

ярлык имя\_структуры; // объявить одну структуру

ярлык имя\_структуры\_1, имя\_структуры\_2; // объявить две структуры

ярлык имя\_структуры[5]; // объявить массив структур

Также есть вариант объявления структуры без ярлыка, т.е. создаём структуру и сразу указываем её имя в конце.

struct {

тип\_данных имя\_переменной\_1;

тип\_данных имя\_переменной\_2;

тип\_данных имя\_переменной\_3;

} имя\_структуры;

Для чего нужны структуры? В большинстве примеров в интернете приводится использование структур для создания базы данных: имя, фамилия, номер телефона, адрес и т.д. Структуры очень удобно передавать по радио и другим интерфейсам связи, на моей практике – в структуре удобнее всего хранить настройки, потому что её можно целиком записать в EEPROM одной строчкой и одной строчкой оттуда же её прочитать, не заморачиваясь с адресами.

**Перечисления** (enum – enumeration) – тип данных, представляющий собой набор именованных констант, нужен для удобства написания программы. Допустим у нас есть переменная mode, отвечающая за номер режима работы устройства. Мы для себя запоминаем, какому значению переменной какой режим будет соответствовать, и где-нибудь себе записываем, например 0 – обычный режим, 1 – режим ожидания, 2 – режим настройки\_1, 3 – режим настройки\_2, 4 – калибровка, 5 – аварийный режим, ошибка. При написании или чтении программы часто придётся обращаться к этому списку, чтобы не запутаться. Можно сделать первый шаг по оптимизации: обозвать каждый режим при помощи константы (дефайна):

#define NORMAL 0

#define WAITING 1

#define SETTINGS\_1 2

#define SETTINGS\_2 3

#define CALIBRATION 4

#define ERROR\_MODE 5

Использование enum ещё немного упрощает эту задачу: перечисление позволяет создать переменную (по умолчанию имеет тип int), которая может принимать только те “названия”, которые для неё указаны. Это удобно также тем, что компилятор выдаст ошибку при создании перечислений с одинаковыми значениями. В случае с дефайнами программа просто будет работать неправильно! Объявление перечисления чем-то похоже на объявление структуры:

enum ярлык {имя1, имя2, имя3, имя4, имя5};

Таким образом мы объявили ярлык. Теперь, используя этот ярлык, можно объявить само перечисление:

ярлык имя\_перечисления;

Также как и у структур, можно объявить перечисление без создания ярлыка (зачем нам лишняя строчка?):

enum {имя1, имя2, имя3, имя4, имя5} имя\_перечисления;

1. **Ход работы**

Ниже представлен интерфейс лабораторной работы:

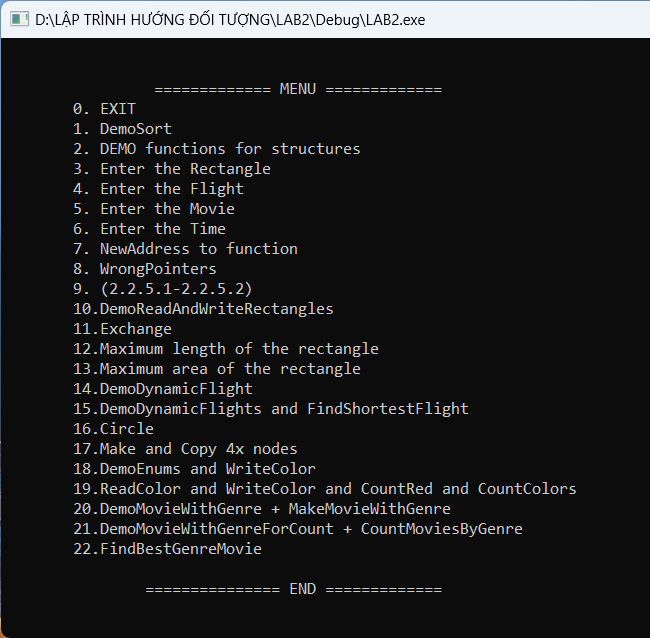


Рисунок 1 – Интерфейс работы

В функции DemoSort() дважды вызовите функцию Sort(), сначала для сount равного 5, а затем равным -1. Убедитесь, что при положительных значениях count сортировка выполняется корректно, а при отрицательных генерируется исключение (Рисунок 2)

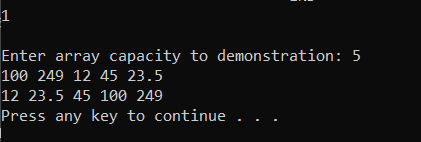


Рисунок 2 – Вывод функции демонстрационной сортировки

В задаче 2 представлена ​​демонстрация функции в структуре (Рисунок 3)

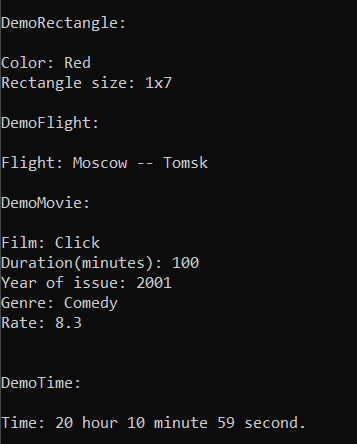


Рисунок 3 – Демонстрационная функция в структуре

В задаче 3 есть демонстрационная функция для ввода прямоугольника (Рисунок 4)

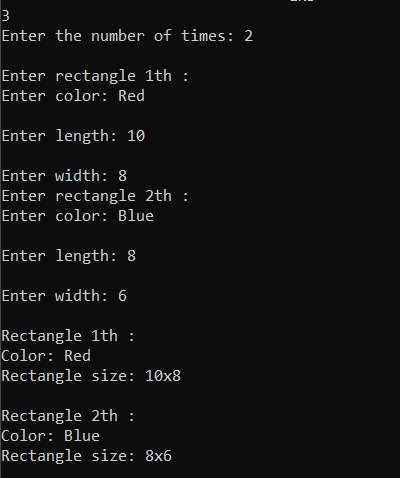


Рисунок 4 – Демонстрация функции прямоугольного ввода

В задаче 4 представлена ​​демонстрация функции ввода полета (Рисунок 5):

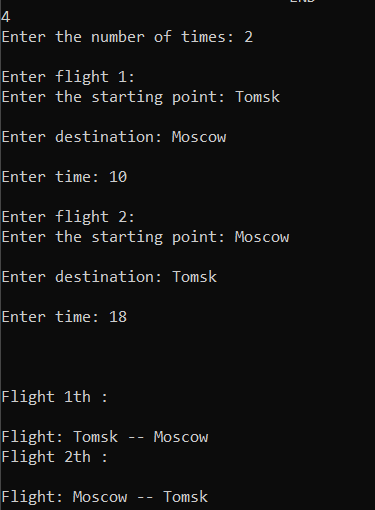


Рисунок 5 – Демонстрация функции ввода полета.

В задаче 5 представлена ​​демонстрация функции ввода информации о фильме. (Рисунок 6):

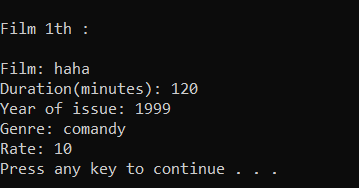


Рисунок 6 – Демонстрация функции ввода информации о фильме.

В задаче 7 представлена ​​демонстрационная функция, которая вводит информацию о фильме и выводит адрес (Рисунок 7)

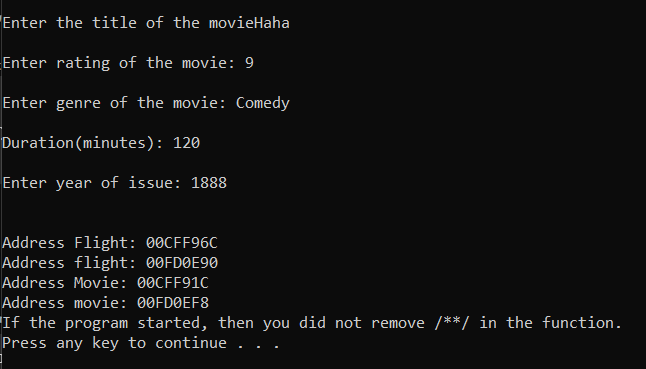


Рисунок 7 – Демонстрация функции

В задаче 8 представлена ​​демонстрационная функция, которая считывает и записывает прямоугольную информацию. (Рисунок 8)

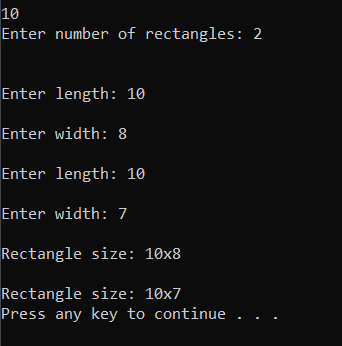


Рисунок 8 – Демонстрация функции считывает и записывает прямоугольную информацию

В задаче 9 представлена ​​демонстрационная функция, которая считывает и записывает информацию о двух прямоугольниках и одновременно изменяет информацию (Рисунок 9)

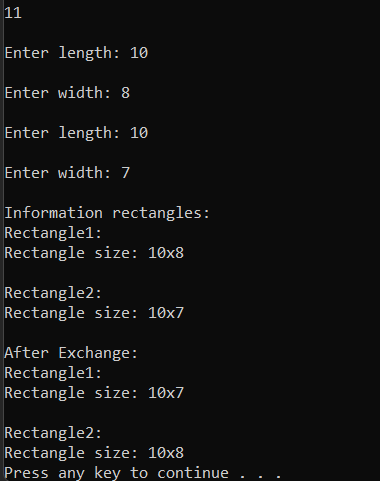


Рисунок 8 – Демонстрационная функция для изменения прямоугольной информации

В задаче 10 есть демонстрационная функция для ввода прямоугольной информации и поиска самого большого прямоугольника (Рисунок 10)

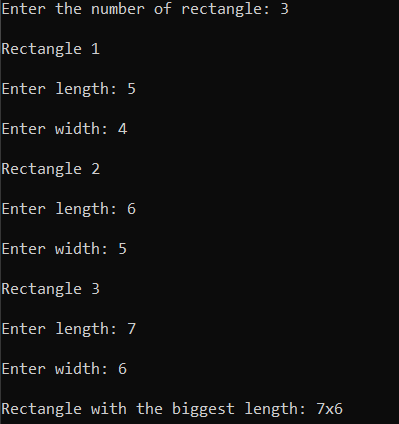


Рисунок 10– Демонстрационная функция для нахождения наибольшей длины

В задаче 11 есть демонстрационная функция для ввода прямоугольной информации и поиска прямоугольника с наибольшей площадью (рис. 11).

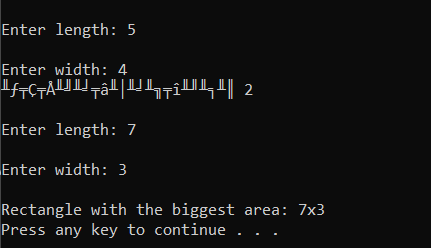


Рисунок 11– Демонстрационная функция для нахождения наибольшей площади

В задаче 12 есть демонстрационная функция для вывода циклической информации (рис. 12).

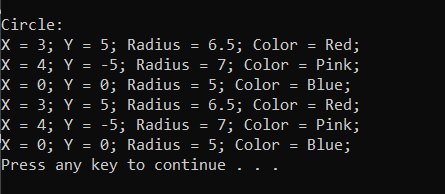


Рис. 12 – Демонстрационная функция для экспорта циклической информации

В задаче 13 есть демонстрация, которая ищет фильмы в списке (рис. 13).

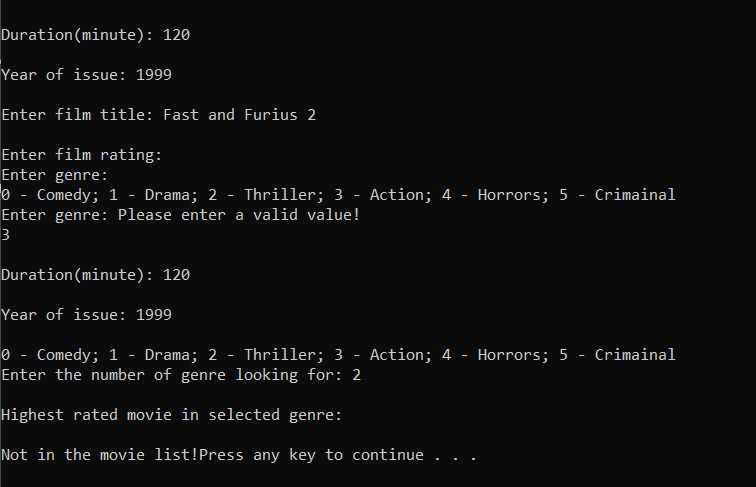


Рис. 13 – Демо поиск фильмов в списке

В задаче 14 есть демонстрация функция инициализирует информацию о рейсах и фильмах, полет. время и создает копии (рис. 14).



Рис. 14 – Демонстрационная функция инициализирует информацию о создании копии

1. **Вывод**

В ходе работы в лаборатории были выполнены все задачи, разработана природа «Структуры и перечисления» в объектно-ориентированном программировании и набор функций для работы с ней.